

# 알튜비튜

## 정수론

오늘은 각종 수의 성질을 다루는 정수론에 대해 배웁니다.  
특히, 최대공약수를 효율적으로 구하는 유클리드 호제법과 소수를 빠른 시간 내에 판별하는 에라토스테네스의 체에 대해 알아보시다.

## 정수론

약수 배수  
최대공약수 최소공배수  
소수 판별

## 최대공약수 구하기

- 소인수분해를 이용하여 구하기
  - 두 수 중 작은 수 기준으로 돌리면서 가장 큰 공통 약수 구하기
- > 구현이 까다로움  
-> 시간복잡도  $O(n)$

## 최대공약수 구하기

- 소인수분해를 이용하여 구하기
  - 두 수 중 작은 수 기준으로 돌리면서 가장 큰 공통 약수 구하기
  - 유클리드 호제법
- > 구현이 까다로움  
-> 시간복잡도  $O(n)$   
-> 시간복잡도  $O(\log(n))$

A와 B의 최대공약수 = A-B와 B의 최대공약수

- $A = a \cdot G$
- $B = b \cdot G$  (a와 b는 서로소)
- $A - B = a \cdot G - b \cdot G = (a - b) \cdot G$   
→ (a - b)와 b 또한 서로소 이므로 A - B와 B의 최대공약수도 G
- $\text{GCD}(A, B) = \text{GCD}(A - B, B) \rightarrow A, B$ 의 차이가 크면 오래 걸림

A와 B의 최대공약수 = A%B와 B의 최대공약수

- $A = a \cdot G$
- $B = b \cdot G$  (a와 b는 서로소)
- $A = q \cdot B + r$  ( $q = A/B$ 의 몫,  $r = A \% B$ )
- $r (A \% B) = a \cdot G - q \cdot b \cdot G = (a - q \cdot b) \cdot G$   
->  $(a - q \cdot b)$ 와 b 또한 서로소 이므로  $A \% B$ 와 B의 최대공약수도 G
- $\text{GCD}(A, B) = \text{GCD}(A-B, B) = \text{GCD}(A-2B, B) = \dots = \text{GCD}(A \% B, B)$

28와 8의 최대공약수 =  $28\%8$ 와 8의 최대공약수

- $28 = 7 \cdot 4$
- $8 = 2 \cdot 4$  (2와 7은 서로소)
- $28 = q \cdot 8 + 4$  ( $q = 12/8$  의 몫,  $4 = 28\%8$ )
- $4 (28\%8) = 7 \cdot 4 - q \cdot 2 \cdot 4 = (7 - q \cdot 2) \cdot 4$   
->  $(7 - q \cdot 2)$  와 2 또한 서로소 이므로  $A\%B(4)$  와  $B(8)$  의 최대공약수도 4
- $\text{GCD}(28, 8) = \text{GCD}(20, 8) = \text{GCD}(12, 8) = \text{GCD}(4, 8)$

## 유클리드 호제법

- 두 정수  $a, b$ 가 주어짐 ( $a > b$ )
- $a$ 와  $b$ 의 최대공약수는  $a \% b$ 와  $b$ 의 최대공약수와 같음
- $a \% b$ 를 구한 후, 왼쪽 값  $>$  오른쪽 값 이어야 하므로  $a \% b$ 와  $b$ 의 위치 바꿈
- $b$ 가 0일 때,  $a$ 의 값이 최대공약수

```
int gcdIter(int a, int b) {  
    while (b) { // b == 0, a가 최대 공약수  
        a %= b; // a = a % b;  
        swap(a, b);  
    }  
    return a;  
}
```



## /<> 2609번 : 최대공약수와 최소공배수 - Silver 5

### 문제

- 두 자연수의 최대공약수와 최소공배수 출력

### 제한 사항

- 입력 범위는  $1 \leq a, b \leq 10,000$

### 풀이

- 최소공배수는  $A * B = G * L$  공식을 이용!

### 예제 입력

24 18

### 예제 출력

6  
72

## 정수론

약수 배수  
최대공약수 최소공배수  
소수 판별

## 소수 구하기

- $2 \sim n$  까지 돌리면서 나눠지는 수가 없는지 판단  $\rightarrow O(n)$
  - $2 \sim \sqrt{n}$  까지 돌리면서 나눠지는 수가 없는지 판단  $\rightarrow O(\sqrt{n})$
- $\rightarrow a = n / (\sqrt{n}\text{이상의 수})$  라는  $a$  가 존재한다면,  $a$  는  $2$  이상  $\sqrt{n}$  이하다 ( $n \% a = 0$ )
- $\rightarrow$  따라서  $\sqrt{n}$  까지만 확인을 하면 그 이상은 확인할 필요가 없다
- (예)  $n = 8, 8 = 2 * 4 = 4 * 2$  로 대칭

## 소수 구하기

- $2 \sim n$  까지 돌리면서 나눠지는 수가 없는지 판단  $\rightarrow O(n)$
- $2 \sim \sqrt{n}$  까지 돌리면서 나눠지는 수가 없는지 판단  $\rightarrow O(\sqrt{n})$

$\rightarrow n$ 이 큰 상황에서  $2 \sim n$  사이에 존재하는 모든 소수를 구해야 한다면?

## 에라토스테네스의 체

- 각 수가 소수인지 판단한 여부를 저장하는 배열 사용
- 2부터 시작해서 해당 숫자의 배수에 해당하는 숫자들을 지워나감 ( $\sim \sqrt{n}$ )
  - > 약수가 존재하면 소수가 아니므로
  - > 해당 숫자는 소수
- $O(n \log(\log n))$ 만에 2 ~ n까지 수에 대해 소수 판정 가능

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 7  | 8  | 9  | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 7  | 8  | 9  | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 7  | 8  | 9  | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |



|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 7  | 8  | 9  | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 7  | 8  | 9  | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 7  | 8  | 9  | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 7  | 8  | 9  | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 |

-> 2, 3, 5, 7, 11, 13, 17, 19, 23, 29

```
void isPrime(int n, vector<bool> &is_prime) {  
    //0과 1은 소수가 아니므로 먼저 제거  
    is_prime[0] = is_prime[1] = false;  
    //2 ~ 제곱근 n까지 검사  
    for (int i = 2; i <= sqrt(n); i++) {  
        if (is_prime[i]) { //i가 소수라면  
            for (int j = i * i; j <= n; j += i) {  
                is_prime[j] = false; //i의 배수를 제거  
            }  
        }  
    }  
}
```

**tip** python3에서는 `math.sqrt()`를 사용해도 되지만, `i ** (1/2)`로 구할 수도 있어요!

## /<> 2960번 : 에라토스테네스의 체- Silver 4

### 문제

- 주어진  $n$  에 대해 에라토스테네스의 체를 적용했을 때,  $k$  번째 지우는 수를 구하는 문제

-> 에라토스테네스의 체 구현하고, 지우는 순서를 카운트하자!

### 제한 사항

- 입력 범위는  $1 \leq K < N \leq 1,000$

예제 입력

7 3

예제 출력

6

예제 입력

15 12

예제 출력

7

예제 입력

10 7

예제 출력

9

## /<> 16563번 : 어려운 소인수분해 - Gold 4

### 문제

- N 개의 자연수 k 를 소인수분해하여  
소인수들을 오름차순으로 출력하는 문제

### 제한 사항

- ✓ N의 범위는  $1 \leq N \leq 1,000,000$
- ✓ k의 범위는  $2 \leq k \leq 5,000,000$

-> 최대 5,000,000까지의 수 중 모든 소수를 구해야하므로 에라토스테네스의 체 활용



## 예제 입력

```
5
5 4 45 64 54
```

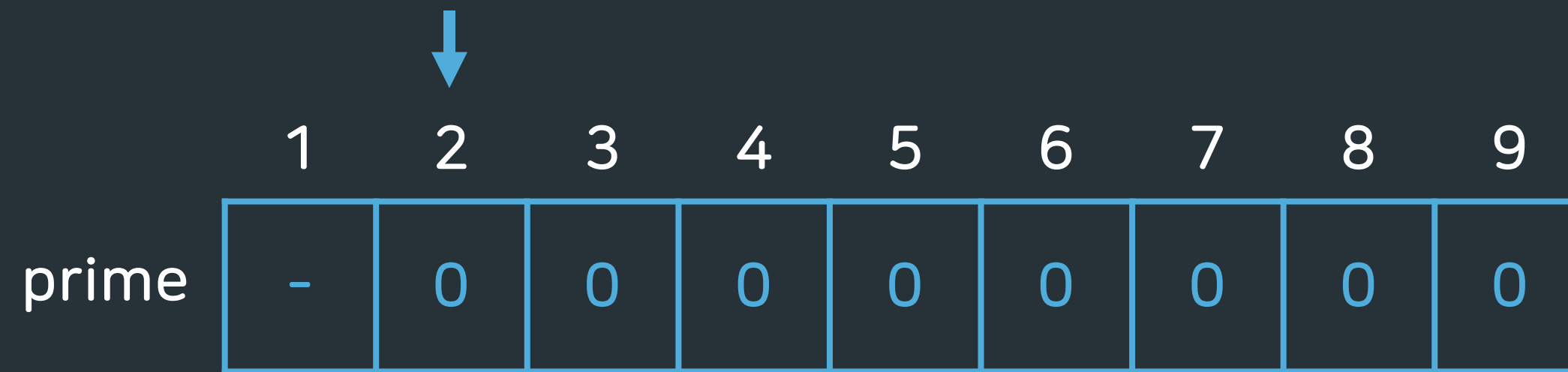
## 예제 출력

```
5
2 2
3 3 5
2 2 2 2 2 2
2 3 3 3
```

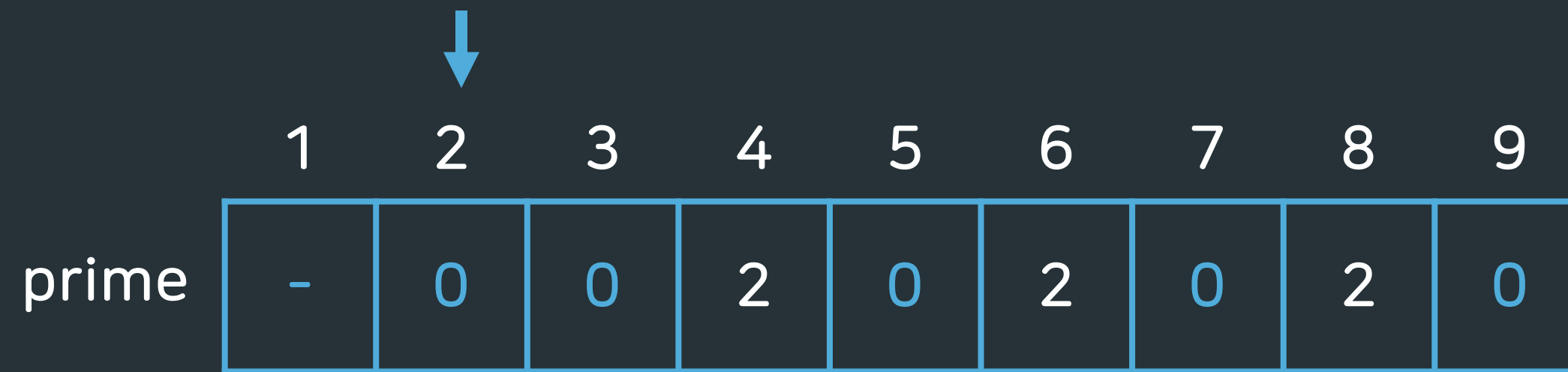
## Hint

1. 에라토스테네스의 체에서 해당 수의 배수를 지워나갈 때 해당 수는 소수였죠…?
2. 구해진 소수를 활용하는 것이 아니라, 이 소수를 구하는 과정을 활용할 수 없을까요?

# 에라토스테네스의 체의 과정 활용?



# 에라토스테네스의 체의 과정 활용?



|       |   |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|---|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| prime | - | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 0 |

- ✓ 소수 판단 여부 정보가 아닌 어느 소수의 배수로 지워졌는지 저장

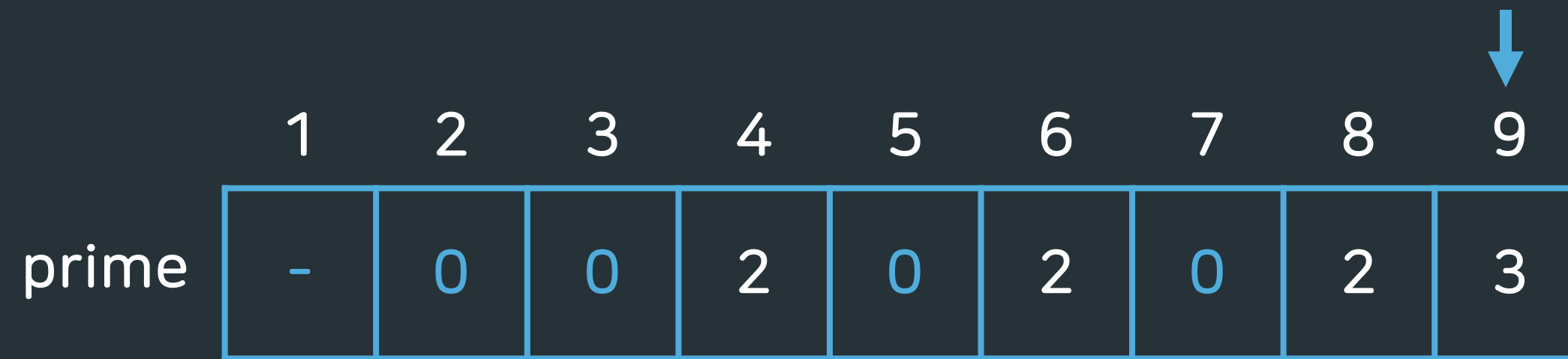
# 에라토스테네스의 체의 과정 활용?

↓

|       |   |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|---|
|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| prime | - | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 3 |

- ✓ 소수 판단 여부 정보가 아닌 어느 소수의 배수로 지워졌는지 저장
- ✓ 이미 값이 존재한다면 갱신 x

# 에라토스테네스의 체의 과정 활용?

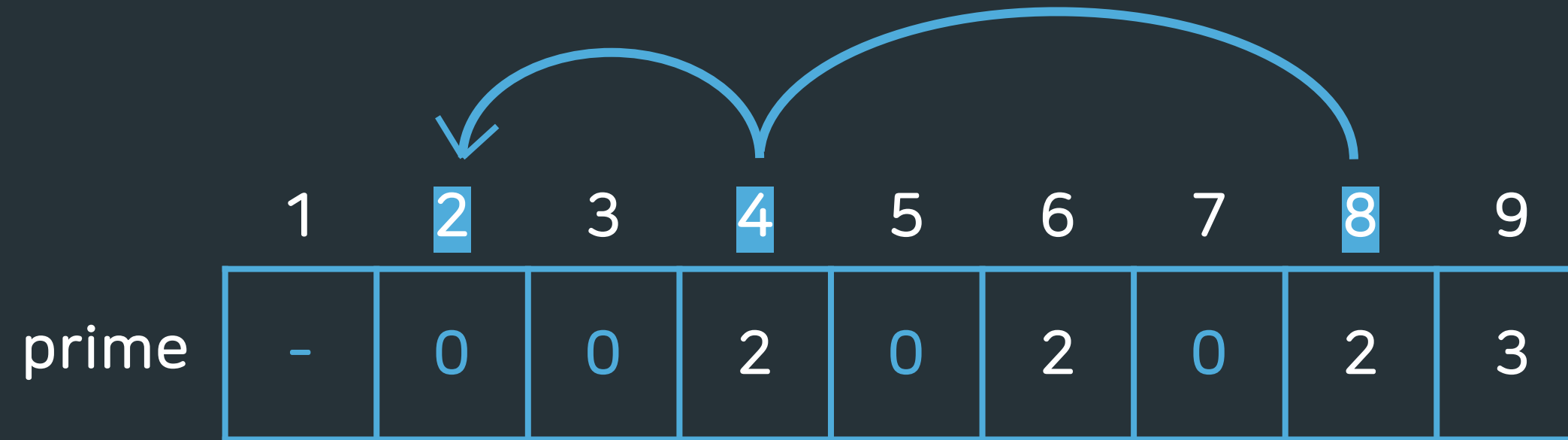


|       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|-------|---|---|---|---|---|---|---|---|---|
| prime | - | 0 | 0 | 2 | 0 | 2 | 0 | 2 | 3 |

- ✓ 소수 판단 여부 정보가 아닌 어느 소수의 배수로 지워졌는지 저장
- ✓ 이미 값이 존재한다면 갱신 x
- ✓ 결국, 해당 인덱스의 가장 작은 소인수가 저장

# 에라토스테네스의 체의 과정 활용?

- (예)  $K = 8$



- ✓ 모두 완료하면 역으로 경로 조사 시작
- ✓  $8 \rightarrow \text{prime}[8] = 2$  (출력)  $\rightarrow 8/2 = 4$
- ✓  $4 \rightarrow \text{prime}[4] = 2$  (출력)  $\rightarrow 4/2 = 2$
- ✓  $2 \rightarrow \text{prime}[2] = 0 \rightarrow$  남은 2 출력 후 종료

## 정리

- 최대공약수를 빠르게 찾는 유클리드 호제법
- 대량의 소수를 빠르게 판별하는 에라토스테네스의 체
- 에라토스테네스의 체는  $2 \sim \sqrt{n}$ 까지 검사 (단, 문제에서 어떻게 활용하느냐에 따라 예외 존재)



## 필수

- /<> 2840번 : 행운의 바퀴 - Silver 4
- /<> 6588번 : 골드바흐의 추측 - Silver 1
- /<> 1735번 : 분수 합 - Silver 3

## 도전

- /<> 9421번 : 소수상근수 - Silver 1
- /<> 2981번 : 검문 - Gold 4

코드리뷰 0 마감 ~ 9월 6일 수요일 18:59

추가제출 마감 ~ 9월 7일 목요일 23:59